Technical Report
CMU/SEI-96-TR-012
ESC-TR-96-012
Carnegie-Mellon University
Software Engineering Institute

Software Risk Management

Ronald P. Higuera
Yacov Y. Haimes

June 1996

Carnegie Mellon University does not discriminate and Carnegie Mellon University is required not to discriminate in admission, employment, or administration of its programs or activities on the basis of race, color, national origin, sex or handicap in violation of Title VI of the Civil Rights Act of 1964, Title IX of the Educational Amendments of 1972 and Section 504 of the Rehabilitation Act of 1973 or other federal, state, or local laws or executive orders.

In addition, Carnegie Mellon University does not discriminate in admission, employment or administration of its programs on the basis of religion, creed, ancestry, belief, age, veteran status, sexual orientation or in violation of federal, state, or local laws or executive orders. However, in the judgment of the Carnegie Mellon Human Relations Commission, the Department of Defense policy of, "Don't ask, don't tell, don't pursue," excludes openly gay, lesbian and bisexual students from receiving ROTC scholarships or serving in the military. Nevertheless, all ROTC classes at Carnegie Mellon University are available to all students.

Inquiries concerning application of these statements should be directed to the Provost, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-6684 or the Vice President for Enrollment, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, telephone (412) 268-2056.

Obtain general information about Carnegie Mellon University by calling (412) 268-2000.

Technical Report
CMU/SEI-96-TR-012
ESC-TR-96-012
June 1996

Software Risk Management

Ronald P. Higuera
Software Risk Management Program
Software Engineering Institute

Yacov Y. Haimes
Center for Risk Management of Engineering Systems
University of Virginia

Risk Program

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This report was prepared for the

SEI Joint Program Office
HQ ESC/ENS
5 Eglin Street
Hanscom AFB, MA 01731-2116

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

FOR THE COMMANDER

Thomas R. Miller, Lt Col, USAF
SEI Joint Program Office

This work is sponsored by the U.S. Department of Defense.

This document is available through Research Access, Inc., 800 Vinial Street, Pittsburgh, PA 15212.
Phone: 1-800-685-6510. FAX: (412) 321-2994. RAI also maintains a World Wide Web home page. The URL is http://www.rai.com

Copies of this document are available through the National Technical Information Service (NTIS). For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. Phone: (703) 487-4600.

This document is also available through the Defense Technical Information Center (DTIC). DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145. Phone: (703) 274-7633.

Contents

List of Figures

Acknowledgements

We are grateful to all the individuals who have contributed over the years to the development of the methodologies, tools, and approaches on software risk management cited and summarized in this paper. In particular we would like to acknowledge the valuable comments and suggestions received from Marvin Carr, Julie Walker, and Bill Wilson on an earlier draft of this paper.

Software Risk Management

Abstract: This paper presents a holistic vision of the risk-based methodologies for Software Risk Management (SRM) developed at the Software Engineering Institute (SEI). SRM methodologies address the entire life cycle of software acquisition, development, and maintenance. This paper is driven by the premise that the ultimate efficacy of the developed methodologies and tools for software engineering is to buy smarter, manage more effectively, identify opportunities for continuous improvement, use available information and databases more efficiently, improve industry, raise the community's playing field, and review and evaluate progress. The methodologies are based on seven management principles: shared product vision, teamwork, global perspective, forward-looking view, open communication, integrated management, and continuous process.

1 Preface

The hierarchy of Software Risk Management (SRM) methodologies discussed in this paper addresses two classes of functions: software acquisition and software development. The basic methodological framework with which functions are managed is composed of the Software Acquisition-Capability Maturity Model (SA-CMM^{SM}) and the Software Capability Maturity Model (SW-CMM^{SM}) and their supporting practices and constructs. This framework for software risk management is supported by three groups of practices:

1. Software Risk Evaluation (SRE)
2. Continuous Risk Management (CRM)
3. Team Risk Management (TRM)

These practices are based on three basic constructs for software risk management developed at the Software Engineering Institute (SEI): Risk Management Paradigm, Risk Taxonomy, Risk Clinic, and Risk Management Guidebooks. The three constructs and three practices will be discussed in subsequent sections.

The complexity of software risk management cannot be understood nor appropriately addressed from the above methodological context alone. To capture the multifarious aspects of this complexity, we make use of hierarchical holographic modeling, where we consider two additional visions or dimensions: the temporal and human dimensions. Thus the three dimensions adopted in this paper to represent the holistic vision of software risk management are the temporal dimension, the methodological dimension and the human dimension.

The temporal dimension is decomposed into two sub-visions:

1. Macro vision represents the global perspective of the acquisition life cycle.
2. Micro vision represents the view of the project manager.

The methodological dimension has already been introduced.

The human dimension addresses the intellectual dimension of software acquisition—the most critical dimension, since software development is such an intellectual activity. Four aspects are identified here:

1. individual
2. team
3. management
4. stakeholder (including customer and client)

The last section shares the experience gained through the deployment of the above methodologies by SEI teams.

Ample literature exists on the process of risk assessment and management. The majority of this literature, however, is devoted to theories and methodologies that have not been subjected to the ultimate test of practice. This paper presents comprehensive theories and processes developed at the SEI at Carnegie Mellon University that have been successfully deployed and tested in the field by numerous clients. (Adhering to confidentiality agreements, the identity of clients will not be revealed.) Authentic statistical information on the use of SEI risk methodologies will be presented and analyzed in the section on the deployment of SEI risk management program.

The goal of SEI Risk Program is to enable engineers, managers, and other decision makers to identify, sufficiently early, the risks associated with software acquisition, development, integration, and deployment so that appropriate management and mitigation strategies can be developed on a timely basis. Time is critical and the goal is to act early before a source of risk evolves into a major crisis. *In other words, being mainly reactive in risk mitigation and control rather than proactive in risk prevention and control is at the heart of good risk management. Furthermore, should the system fail regardless of all risk management efforts, then ensuring the safe failure (e.g., safe shutdown) of the system must be the mandate of the software risk manager.* Clearly, the secret to effective risk management is the trade-off of mitigation cost against the potential adverse effects of avoided risk. In this context, the value of the methodologies and tools for software risk management is to buy smarter, manage more effectively and identify opportunities for continuous improvement, use available information and databases more efficiently, improve industry and raise the community's playing field, and review and evaluate the progress made on risk management.

It is important to note that the developed software risk methodologies have three fundamentally different, albeit complementary, objectives:

1. risk prevention
2. risk mitigation and correction
3. ensuring safe system failure

The following seven risk management principles are instrumental in the quest to achieve these three objectives [Higuera 94]:

Shared product vision

- sharing product vision based upon common purpose, shared ownership, and collective commitment
- focusing on results

Teamwork

- working cooperatively to achieve a common goal
- pooling talent, skills, and knowledge

Global perspective

- viewing software development within the context of the larger system-level definition, design, and development
- recognizing both the potential value of opportunity and the potential impact of adverse effects, such as cost overrun, time delay, or failure to meet product specifications

Forward-looking view

- thinking toward tomorrow, identifying uncertainties, anticipating potential outcomes
- managing project resources and activities while anticipating uncertainties

Open communication

- encouraging the free flow of information between all project levels
- enabling formal, informal, and impromptu communication
- using consensus-based process that values the individual voice (bringing unique knowledge and insight to identifying and managing risk)

Integrated management

- making risk management an integral and vital part of project management
- adapting risk management methods and tools to a project's infrastructure and culture

Continuous process

- maintaining constant vigilance
- identifying and managing risks routinely throughout all phases of the project's life cycle

2 Introduction

Making informed decisions by consciously assessing what can go wrong, as well as the likelihood and severity of the impact, is at the heart of risk management. Making informed decisions involves the evaluation of the trade-offs associated with all policy options for risk mitigation in terms of their costs, benefits, and risks, and the evaluation of the impact of current decisions on future options. This process of risk management embodies the identification, analysis, planning, tracking, controlling, and communication of risk.

Acquisition, development, and deployment programs continue to suffer large cost overruns, schedule delays, and poor technical performance. Generally, this is a result of failing to deal appropriately with uncertainty in the acquisition and development of complex, software-intensive and software-dependent systems. The acquisition and development communities, both governmental and industrial, lack a systematic way of identifying, communicating, and resolving technical uncertainty. Often the focus is on the symptoms of cost overruns and schedule delays rather than on the root causes in product acquisition and development. In fact, all areas in systems development are potential sources of software risks (see Figure 1) since it involves technology, hardware, software, people, cost, and schedule.

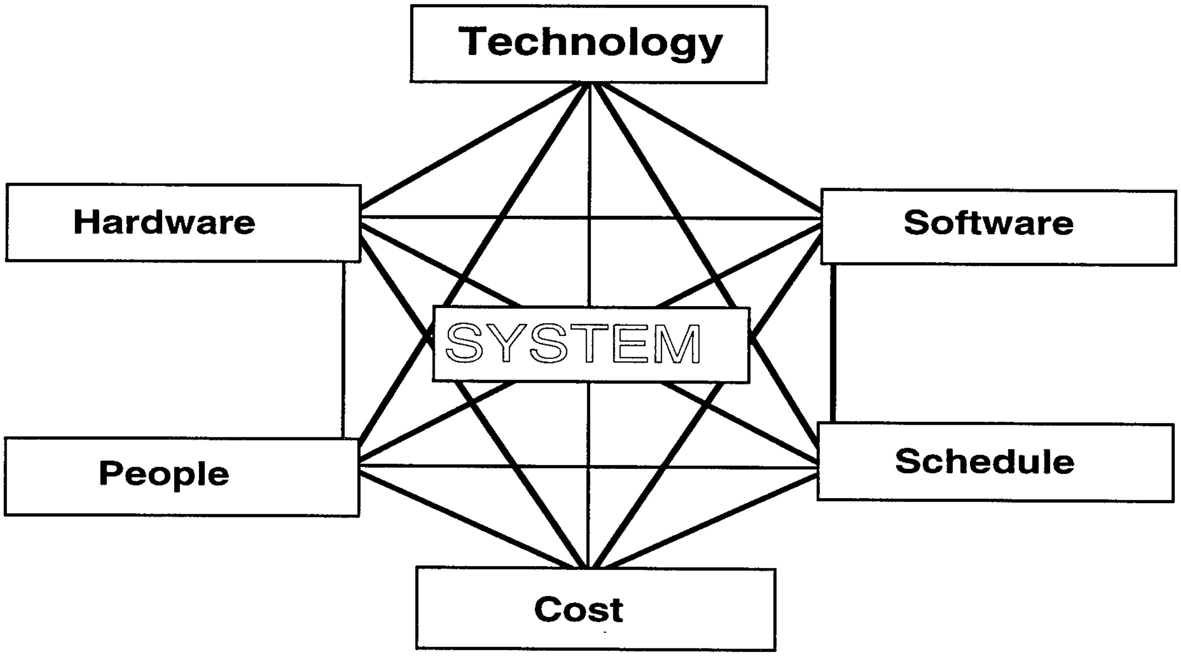


Figure 1: Risks Within a System Context

Risk is commonly defined as a measure of the probability and severity of adverse effects [Lowrance 76]. *Software technical risk* can be defined as a measure of the probability and severity of adverse effects inherent in the development of software that does not meet its intended functions and performance requirements [Chittister 93].

The need to manage risk increases with system complexity. Figure 2 demonstrates this concept by indicating that as the complexity of the system increases, both technical and non-technical (cost and schedule) risks increase. There is an increasing need for more systematic methods and tools to supplement individual knowledge, judgment, and experience. These human traits are often sufficient to address less complex risks. It is worth noting that many managers believe that they are managing risk in its multifaceted dimensions. The fact of the matter is that they are merely managing cost and schedule along with isolated cases of technical risk. The SEI Risk Program provides a structured process, supported by methods and tools, for identifying, analyzing, and mitigating the uncertainties encountered in a specific software engineering effort. Many of the most serious issues encountered in system acquisition are the result of risks that either remain unrecognized and/or are ignored until they have already created serious consequences. This focus on risk management is important because structured techniques, even quite simple ones, can be effective in identifying risk, and approaches, procedures, and techniques do exist for risk mitigation.

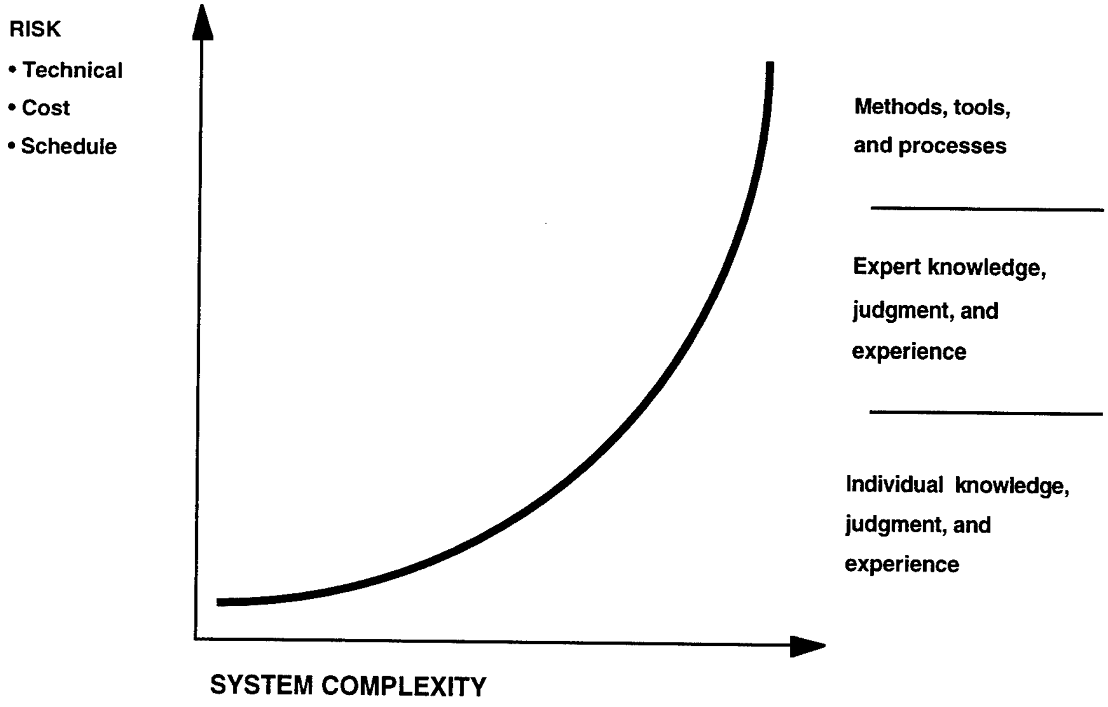
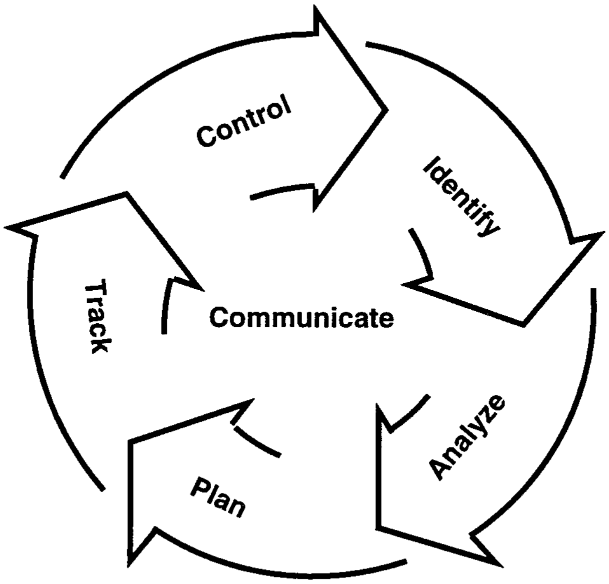


Figure 2: The Need to Manage Risk Increases With System Complexity

Experience has shown that only a few programs are managing risk in a systematic way, and that the approaches of the programs that do manage risk tend to be *ad hoc*, undocumented, and incomplete [Kirkpatrick 92]. SEI teams have also found that software risk is among the least measured or managed in a system.

In its attempt to respond to these problems, the goal of the SEI Risk Program is to improve the process for acquisition and development of software-intensive systems. In particular, its aims are: to enable acquisition and development managers and engineers to make better decisions (by identifying risk before they become problems); to communicate risks in a positive, non-threatening way; and to resolve technical risk in a cost-effective manner. The three groups of methodologies (SRE, TRM, and CRM) are based on three basic constructs for risk management developed at the SEI. These are: Risk Management Paradigm, Risk Taxonomy, and Risk Clinic. These constructs, the three groups of methodologies cited above, and the two methodological frameworks will be discussed in subsequent sections. The Risk Management Paradigm (Figure 3), which advocates a continuous set of activities to identify, confront, and resolve technical risk [Van Scoy 92], will be further discussed in Section 4.1.1.



A continuous set of activities to identify, confront, and resolve technical risk

Figure 3: SEI Risk Management Paradigm

3 A Holistic Vision of Software Risk Management

The complex process of software acquisition encompasses most, if not all, aspects associated with software risk management. Thus, it seems natural to focus on the entire life cycle of the software acquisition process in developing a holistic vision of risk management. Indeed, risk management of software engineering cannot be restricted to any subset or a single phase of the life cycle of software development.

The following objectives of the overall methodological framework for software risk management apply to software-intensive systems.

1. Improve the process of software acquisition in organizations.
2. Improve software risk management methodology, technology, and practice in the acquisition process.
3. Improve the access to, acquisition, repository, use, and integration of information and data for software acquisition in industry and government.
4. In general, institutionalize risk management and decision support within the software acquisition community and make it an integral part of the community's practice.

The multifarious and complex nature of the acquisition process is a fundamental attribute of software-intensive systems. This complex process involves multiple decision makers and multiple non-commensurate objectives, a multitude of sources of risks and uncertainties, and an evolving technology that is shifting the focus from hardware to software. Furthermore, software is playing an increasingly central and pivotal role in systems integration. This complexity of software-intensive systems makes modeling and managing of such systems more challenging and demands new approaches and new schemes. Thus, the representation of all aspects and perspectives of software risk management in a single model, or paradigm, is impractical. The multifaceted dimensions of the risks associated with the software acquisition process cannot be modeled or described by one single vision or a single planar model, and any attempt to do so would necessarily compromise the intended communication between that limited description and the reader. No single planar picture of a car, for example, would be able to communicate all the intricate functions and components of this complex system. The same holds true for the methodological framework for software risk management developed at the SEI. Here, one may distinguish among at least three visions: temporal, methodological, and functional. In this paper, we make use of hierarchical holographic modeling (HHM) [Haimes 81] to construct a holistic vision that represents the software risk management process.

3.1 Temporal Dimension

It is plausible to assert that the genesis of a formal acquisition process can be traced to the Statement of Needs and Requirements. In terms of risk management, the seeds of critical sources of risk are often sown at this seemingly benign stage. An example from urban development demonstrates this point. A mayor and the city council identify a need for a new housing development. Given the high cost of land due to its scarcity, the requirements for meeting these needs evolve into the construction of high rise apartments. At this stage, the risks that the new project might become a major slum and a magnet for crime and drug distribution are not considered. The goal of risk management is in the prevention of such risks. The importance the Needs and Requirements stage places this stage at the foundation of the holistic vision of software risk management depicted in Figure 5, which follows the introduction of all components of the hierarchical holographic model for software risk management.

The total acquisition life cycle is presented in two separate yet overlapping visions. The *micro vision* primarily represents the view of the *project manager.* The *macro vision* represents the more global and broader perspective of the *acquisition life cycle.* It is worth noting that within each stage of the temporal domain, the human dimension (individual, team, manager, or stakeholder) has a different and unique role to play.

Micro vision

1. specification
2. solicitation (including request for proposal and contractor selection)
3. design and development (including architecture)
4. systems integration (including deployment and maintenance)

Macro vision

1. conceptual design
2. demonstration/validation
3. engineering, manufacturing, development, and production
4. maintenance and major upgrade (including termination)

Although the two perspectives somehow overlap, they do represent the life cycle development of software in its multifaceted dimensions. For example, most software risk-based methodologies developed so far are applicable to the developmental stages identified within the micro vision, because most managerial decisions are indeed made in this domain. At the same time, however, the only way that the micro vision makes sense is when it is understood and acted upon from the broader macro vision.

Because the Needs and Requirements stage is too important a contributor to the sources of risk, it is separated in our overall model presentation from the micro and macro visions. Indeed, the Needs and Requirements stage constitutes the base of the spiral model depicted in Figure 5. One reason that many of the seeds of risk are sown during the Needs and Requirements stage is that software engineering remains more an art than a science in spite of the major gains that have materialized during the last several years. It is worth noting the testimony by William Wulf before a Congressional committee in 1989 when he served as Assistant Director for Computer and Information Science and Engineering at the National Science Foundation. Commenting on the need to improve our knowledge on software, Wulf said, "The fundamental intellectual foundation, even the appropriate mathematics, does not exist" to solve "software crises" [House 89]. Even earlier, in 1987, Frederick P. Brooks recognized the quintessential role that the Needs and Requirement stage plays in software risk:

> *The hardest single part of building a software system is deciding precisely what to build. No other part of the work so cripples the resulting system if done wrong; no other part is more difficult to rectify later.*

Therefore, the most important function that the software builder performs for the client is the iterative extraction and refinement of the product requirements. For the truth is that the client rarely knows what he or she wants. The client usually doesn't know what questions must be answered, and he or she probably hasn't thought of the problem in the detail necessary for specification.

Clearly, understanding and appreciating the evolution of risks during the temporal life cycle are requisites for effective risk management.

3.2 Methodological Dimension

The risk-based methodologies discussed in this paper are designed to improve the overall software developmental process an offer a fresh way to integrate knowledge into the software acquisition process in a way that would enable managers to make more timely decisions. This is accomplished by providing a structured approach to the assessment and management of the risks and uncertainties associated with the developmental process. In risk assessment the analyst often attempts to answer the following three questions: What can go wrong? What is the likelihood that it would go wrong? and What are the consequences? [Kaplan 81] Answers to these questions help risk analysts identify, measure, quantify, and evaluate the consequences and impacts of risks. The remaining risk analysis builds on the risk assessment process by seeking answers to a second set of questions: What can be done? What options are available? What are their associated trade-offs in terms of all costs, benefits, and risks? and What are the impacts of current management decisions on future options [Haimes 91]? Only when these questions are addressed in the broader context of management can total risk management be realized. The methodologies developed at SEI provide answers to these sets of questions.

More specifically, these methodologies provide answers to the following sample of more specific questions:

> *I know that improving the process will improve my software. How do I choose the improvement method that will have the most effect for my current state? How do I secure against major disasters? What cost will I face?*

What makes a good software professional? How can I inspire my team to their best efforts? How do I know training is of any use? How can I convince my management to invest in risk management? How can I overcome resistance to change?

How do I make trade-offs among the risk factors affecting software quality, cost overrun, and time delay in project completion schedule?

The hierarchy of SRM methodologies discussed in this paper addresses the two life cycle functions of software acquisition and development. The basic methodological framework with which the functions are managed is composed of the SW-CMM^{SM}, and the SA-CMM^{SM}.

The above methodological framework for software risk management is supported by three groups of practices:

1. SRE
2. CRM
3. TRM

These practices build on three basic constructs of risk management:

1. the Risk Management Paradigm
2. the Risk Taxonomy
3. the Risk Clinic

Each will be discussed in detail in Section 4. Figure 4 depicts the relationships among the currently available models, practices, and constructs for risk management of software-intensive systems.

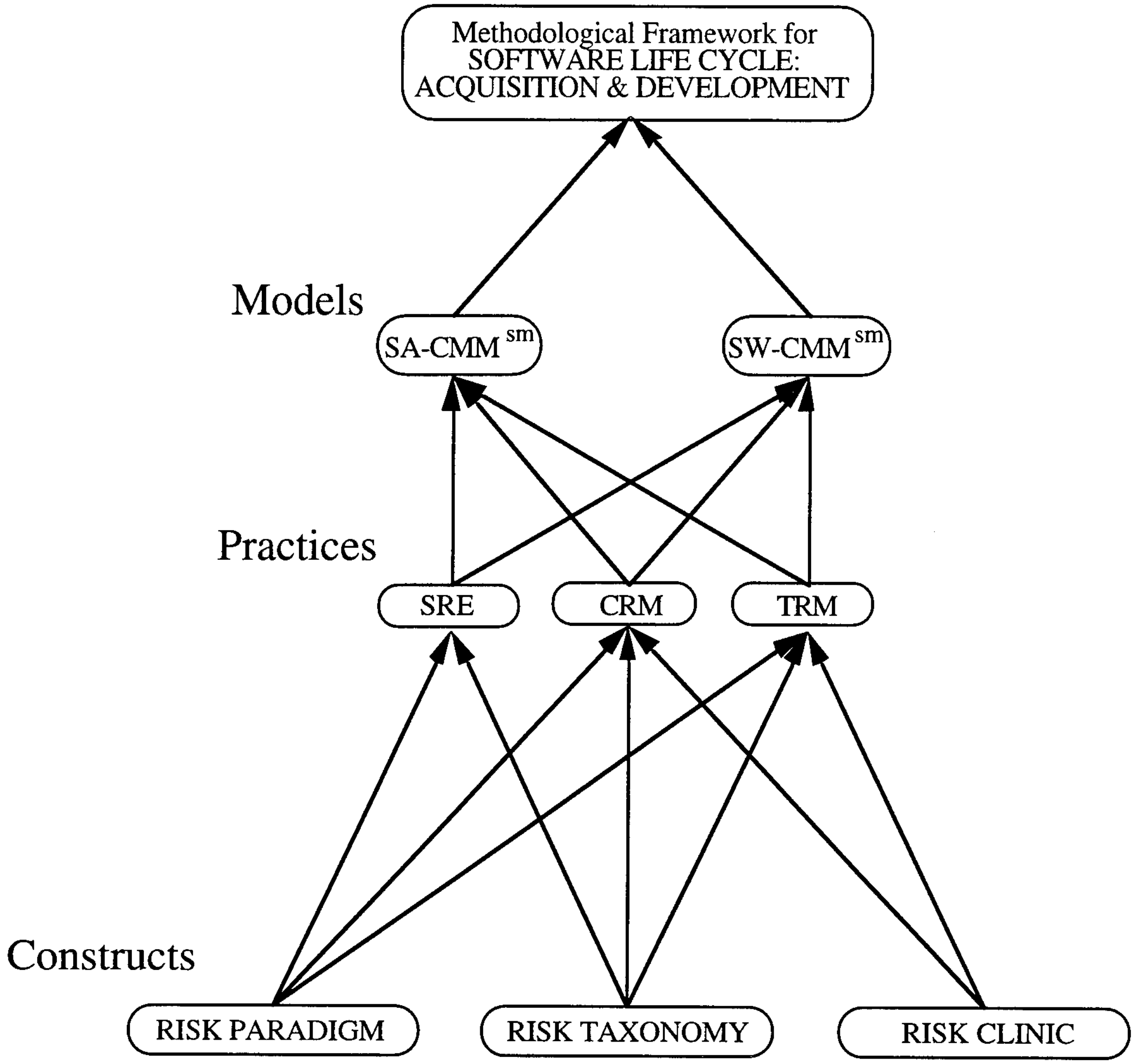


Figure 4: Methodological Framework for Software Risk Management

3.3 Human Dimension

The third dimension in the holistic vision of software risk management addresses the intellectual dimension of software acquisition—the most critical one, since software development is such an intellectually intensive activity. Four perspectives are identified:

1. individual
2. team

3. management
4. stakeholder

There is an obvious interplay and overlap among all four elements that constitute the human dimension. The individual perspective represents an important source of risk in software development. The lack of training, knowledge, skill, commitment to the project, loyalty to the organization as a whole, dedication to quality, and many other factors are critical to the initiation of risks and to their identification at an early stage of the development.

Although teams are composed of individuals, the team perspective is different from the Individual one. In their book, *The Wisdom of Teams,* Katzenbach and Smith [Katzenbach 93] provide the following succinct definition of a team:

> *A team is a small number of people with complementary skills who are committed to a common purpose, performance goals, and approach for which they hold themselves mutually accountable.*

It is clear from the above definition that software risk and its management are heavily dependent on the quality of the team and its commitment to identifying, preventing, and managing software risk. In the subsequent section, we will elaborate on the role of teams in risk management through a discussion of TRM methodology.

The third perspective of the human dimension is management. These are the engineers who are managers of risk, and risk experts who are managers of engineering systems. Furthermore, one must also appreciate the hierarchical managerial structure and the consequences of its divisions [Chittister 94]:

> *Upper management views risk almost exclusively in term of profitability, schedule, and quality. Risk is also viewed in terms of the organization as a whole and the effects on multiple projects or a product line.*
>
> *Program management is concerned with profitability. It concentrates more on cost, schedules, product specificity, quality, and performance, usually for a specific program or project.*
>
> *The technical staff overlaps with the individual element, and may have some of its members in supervisory roles. This group concerns itself primarily with technical details of components, subassemblies, and products for one or more projects.*

Clearly, differences among the risk managers at each level of this hierarchical decision-making structure are caused by numerous factors, including the scope and level of responsibilities, time horizon, functionality, requirements of skill, knowledge, and expertise.

The stakeholders—the fourth perspective in the human dimension—are also a conglomerate of constituencies. This may include government agencies, a specific branch of the Armed Forces, major corporations, and other power brokers—all of whom have direct or indirect interest in the acquisition of software.

Understanding the role of each and all of the four perspectives in the human dimension is essential to effectively assessing and managing software risk.

3.4 Graphic Representation of the Holistic Vision of Software Risk Management

The holistic vision of software risk management is depicted in Figure 5, where the three dimensions of the software acquisition process—temporal, methodological, and human dimensions—are represented in a spiral that evolves upward over time. The spiral mode emphasizes the iterating nature of risk management, where at each stage of the software acquisition process, the manager-analyst adheres to the Risk Paradigm—identify, analyze, plan, track, and control [Van Scoy 92]. Communication is, of course, at the heart of the Risk Paradigm.

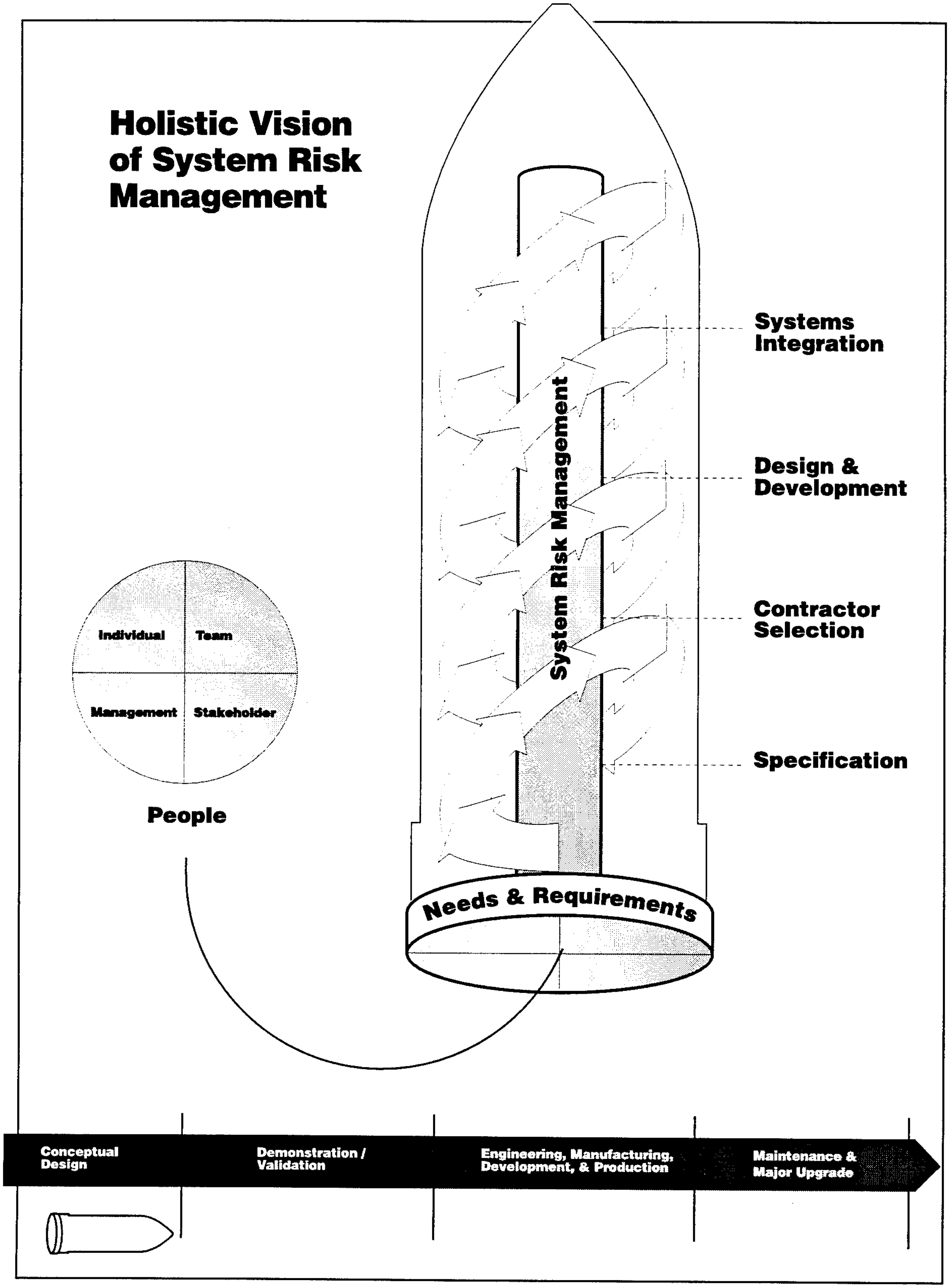


Figure 5: Holistic View of Risk Management

The micro level of the temporal dimension is represented by its four stages, evolving upward in the spiral (specification; contractor selection; design and development; and systems integration). The four stages of the macro level (conceptual design; demonstration/ validation; engineering, manufacturing, development, and production; and maintenance and major upgrade) are depicted on the horizontal line of Figure 5, representing the time element that characterizes the macro level.

The methodologies associated with system risk management are presented in Figure 5 through the rising column within the upward evolving spiral. The intention is to emphasize the fact that at each stage of the software acquisition decision-making process, the manager-analyst is able to make use of these methodologies.

The third dimension—human—is represented through a cross section of the upward evolving temporal domain. At each stage of the acquisition process, the influence, involvement, leadership, imagination, and impact of the individual, the team, the management, and the "external" stakeholders are felt.

The hierarchical holographic model that represents the holistic vision of software risk management as depicted in Figure 5 will be revisited in this paper after each of the three dimensions has been discussed in some detail.

4 Software Risk Management Methodologies

Although the Risk Paradigm is not considered a "methodology" per se, it is discussed under the methodological dimension. The Risk Paradigm transcends all risk analysis activities discussed earlier; for this reason, it constitutes the foundation of each stage in the spiral form depicted in Figure 5. Similar reasoning applies to the Risk Taxonomy [Carr 93] and to the Risk Clinic. The taxonomy provides a framework for organizing and studying the breadth of software development issues and hence provides a structure for surfacing and organizing software development risks. Since several of the methodologies discussed here make use of the Risk Taxonomy, it is presented along with the risk management paradigm as "Basic Constructs to Risk Management." The Risk Clinic is a workshop that constitutes an important part of CRM and TRM.

4.1 Basic Constructs to Risk Management

Three basic constructs will be discussed here. All three constructs build on the seven risk management principles discussed in the preface—shared product vision, teamwork, global perspective, forward-looking view, open communication, integrated management, and continuous process.

4.1.1 Risk Management Paradigm

The risk management paradigm (see Figure 3) depicts the different activities involved in the management of risk associated with software development [Van Scoy 92]. The paradigm is represented by a circle to emphasize that risk management is a continuous process, while the arrows show the logical flow of information between the activities. Communication is placed in the center of the paradigm because it is both the conduit through which all information flows and, often, the largest obstacle in risk management. Essentially, the paradigm is a framework for software risk management. From this framework, a project may structure a risk management practice best fitting into its project management structure. A brief summary of each risk management paradigm activity is described below.

Identify

Before risks can be managed, they must be identified. Identification surfaces risks before they become problems. The SEI has developed techniques for surfacing risks by the application of a systematic process that encourages project personnel to raise concerns and issues. One such method, the SRE, is described in a subsequent section.

Analyze

Analysis is the conversion of risk data into risk decision-making information. Analysis provides the basis for the project manager to work on the "right" and most critical risks.

Plan

Planning turns risk information into decisions and actions. Planning involves developing actions to address individual risks, prioritizing risk actions, and creating an integrated risk management plan. The plan for a specific risk can take many forms. For example:

- Mitigate the impact of the risk by developing a contingency plan (along with an identified triggering event) should the risk occur.
- Avoid a risk by changing the product design or the development process.
- Accept the risk and take no further action, thus accepting the consequences if the risk occurs.
- Study the risk further to acquire more information and better determine its characteristics to enable wiser decision-making.

The key to risk action planning is to consider the future consequences of a decision made today.

Track

Tracking consists of monitoring the status of risks and the actions taken to ameliorate them. Appropriate risk metrics are identified and monitored to enable the evaluation of the status of as well as of risk mitigation plans. Tracking serves as the "watchdog" function of management.

Control

Risk control corrects deviations from planned risk actions. Once risk metrics and triggering events have been chosen, there is nothing unique about risk control. Risk control melds into project management and relies on project management processes to control risk action plans, corrects for variations from plans, responds to triggering events, and improves risk management processes.

Communicate

Risk communication lies at the center of the model to emphasize both its pervasiveness and its criticality. Without effective communication, no risk management approach can be viable. While communication facilitates interaction among the elements of the model, there are higher level communications to consider as well. In order to be analyzed and managed correctly, risks must be communicated to and between the appropriate organizational levels. This includes levels within the development project and organization, within the customer organization, and most especially, across that threshold between the developer, the customer, and, where different, the user. Because communication is pervasive, our approach is to address it as integral to every risk management activity and not as something performed outside of, or as a supplement to, other activities.

4.1.2 Risk Taxonomy

The Risk Taxonomy follows the life cycle of software development and provides a framework for organizing data and information. The taxonomy-based identification method provides the organization developing software with a systematic interview process with which to identify sources of risk.

The taxonomy construct consists of a Taxonomy-Based Questionnaire and a process for its application. The taxonomy organizes software development risks into three levels: class, element, and attribute. The questionnaire consists of questions under each taxonomic attribute that are designed to elicit the range of risks and concerns potentially affecting the software product. The application process is designed such that the questionnaire can be used in a practical and efficient manner consistent with the objective of surfacing project risks. Both the questionnaire and the application process have been developed using extensive expertise and multiple field tests.

The taxonomy methodology [Carr 93] is an instrument with which one can obtain a broad, system level of risks. These risks are commonly identified by program members, and are classified by categories within the hierarchical structure of the taxonomy. Moreover, the taxonomy identifies risk areas for more detailed investigation and is applied by interviewing peer groups of managers, engineers, and support personnel. Figure 6 and Figure 7 depict the hierarchical nature of the taxonomy.

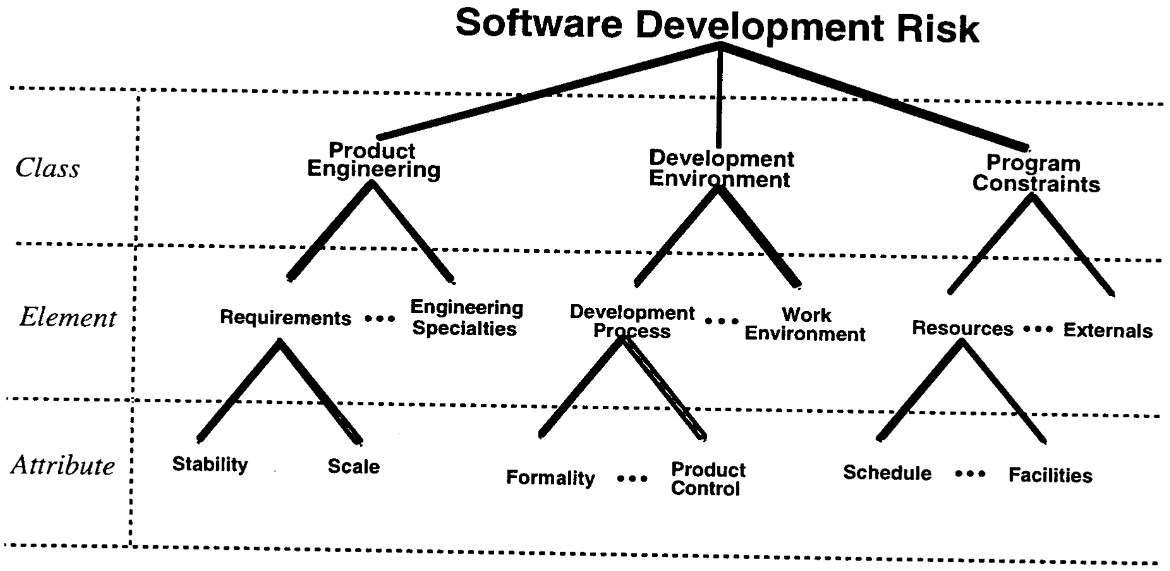


Figure 6: Complete Taxonomy

A. Product Engineering
   1. Requirements
      a. Stability
      b. Completeness
      c. Clarity
      d. Validity
      e. Feasibility
      f. Precedent
      g. Scale
   2. Design
      a. Functionality
      b. Difficulty
      c. Interfaces
      d. Performance
      e. Testability
      f. Hardware Constraints
      g. Non-Developmental Software
   3. Code and Unit Test
      a. Feasibility
      b. Testing
      c. Coding/Implementation
   4. Integration and Test
      a. Environment
      b. Product
      c. System
   5. Engineering Specialties
      a. Maintainability
      b. Reliability
      c. Safety
      d. Security
      e. Human Factors
      f. Specifications

B. Development Environment
   1. Development Process
      a. Formality
      b. Suitability
      c. Process Control
      d. Familiarity
      e. Product Control
   2. Development System
      a. Capacity
      b. Suitability
      c. Usability
      d. Familiarity
      e. Reliability
      f. System Support
      g. Deliverability
   3. Management Process
      a. Planning
      b. Project Organization*
      c. Management Experience
      d. Program Interfaces
   4. Management Methods
      a. Monitoring
      b. Personnel Management*
      c. Quality Assurance
      d. Configuration Management
   5. Work Environment
      a. Quality Attitude*
      b. Cooperation*
      c. Communication
      d. Morale*

C. Program Constraints
   1. Resources
      a. Schedule
      b. Staff
      c. Budget
      d. Facilities
   2. Contract
      a. Type of Contract*
      b. Restrictions
      c. Dependencies
   3. Program Interfaces
      a. Customer*
      b. Associate Contractors
      c. Subcontractors*
      d. Prime Contractor*
      e. Corporate Management
      f. Vendors
      g. Politics*

* Areas in which risks are not expected to be encountered prior to contract award

Figure 7: Taxonomy of Software Risks: Overview

The SEI taxonomy of software development maps the characteristics of software development and software development risks. The questionnaire is a list of non-judgmental questions to elicit issues, concerns (i.e., potential risks), and risks in each taxonomic group. Hence, the questionnaire ensures that all risk areas are systematically addressed, while the application process is designed to ensure that the questions are asked of the right people and in the right manner to produce optimum results.

The questionnaire application is semi-structured. The questions and their sequence are used as a defining but not as a limiting instrument. That is, the questions are asked in a given sequence, but the discussion is not restricted to that sequence. This is done to permit context and culture-sensitive issues to arise. A completely structured interview, while arguably yielding more reliable data for subsequent analysis across different projects, may also yield less valid data. Since the pragmatics of risk management are paramount, the semi-structured format was chosen by the SEI. In other words, the questionnaire can be described as a form of structured brainstorming.

The taxonomy's risk identification method identifies and clarifies the uncertainties and concerns of a project's technical and managerial staff. The software taxonomy is organized into three major classes:

1. *product engineering:* the technical aspects of the work to be accomplished
2. *development environment:* the methods, procedures, and tools used to produce the product
3. *program constraints:* the contractual, organizational, and operational factors within which the software is developed, but which are generally outside of the direct control of the local management

These taxonomic classes are further divided into *elements* and each element is characterized by its *attributes.*

4.1.3 Risk Clinic

A Risk Clinic is a workshop that takes the SEI CRM and TRM and adapts and integrates it with a client's communication channels, infrastructure, existing practices, project management, risk management (if any), and technical problem management (see Figure 8).

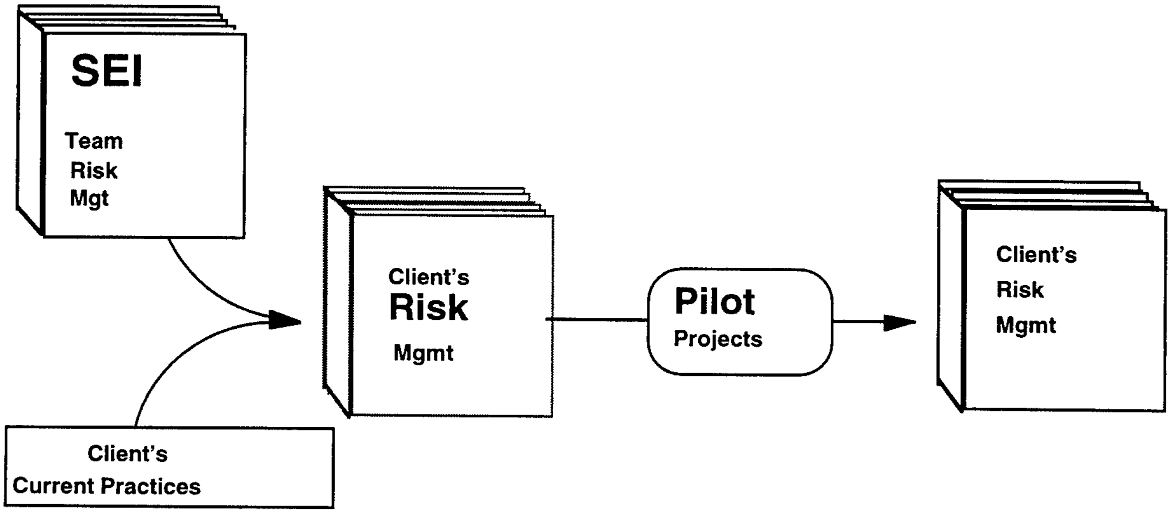


Figure 8: Risk Clinic Integrates Risk Management with Current Practices

The Risk Clinic is the cornerstone of a process of interactive, adaptive transition that spans several months. It takes place after planning meeting is held between SEI and the client to establish the schedules deliverables and identify the pilot projects. The Risk Clinic is the centerpiece of the transition effort and should occur within 30 days of the planning meeting to keep up the momentum. If more than one pilot project is considered, multiple Risk Clinics should be held to provide the chance to evaluate alternative types of activities and procedures. The most successful can then become part of the client's risk management practice or they can be used to provide alternatives for the organization. The executive briefing is an internal briefing used by the client sponsors to educate management and pilot project personnel about the risk management transition effort.

Once the Risk Clinic has established proposed client risk management practices, these are implemented into one or more pilot projects whose progress is followed with coaching meetings between SEI and pilot project personnel. These meetings are used not only to evaluate progress, but also to adjust and revise practices. Coaching continues until the proposed risk management practice has been fully implemented and tested. Revisions and changes are then made to improve the client's risk management practices and these improvements are documented so as to institutionalize them.

Figure 9 illustrates the overall process for conducting a Risk Clinic. A typical Risk Clinic takes two full days of high-energy activity with personnel from the SEI, from the client's pilot project, and from the process improvement or process definition groups (typically the software engineering process group). Once the client's proposed risk management framework has been established, a plan for incremental transition and implementation of these activities is defined.

Optional preliminary surveys of change implementation history and the cultural barriers to change[1] can be used to help identify any specific barriers to change that may need to be overcome during this transition, as well as to identify any enablers for change that can be used to aid the transition. Specific milestones and target dates are identified for the next several months (e.g., implement a risk database, complete with all report templates, in three months). Finally, the transition plan is discussed and a preliminary agenda is defined.

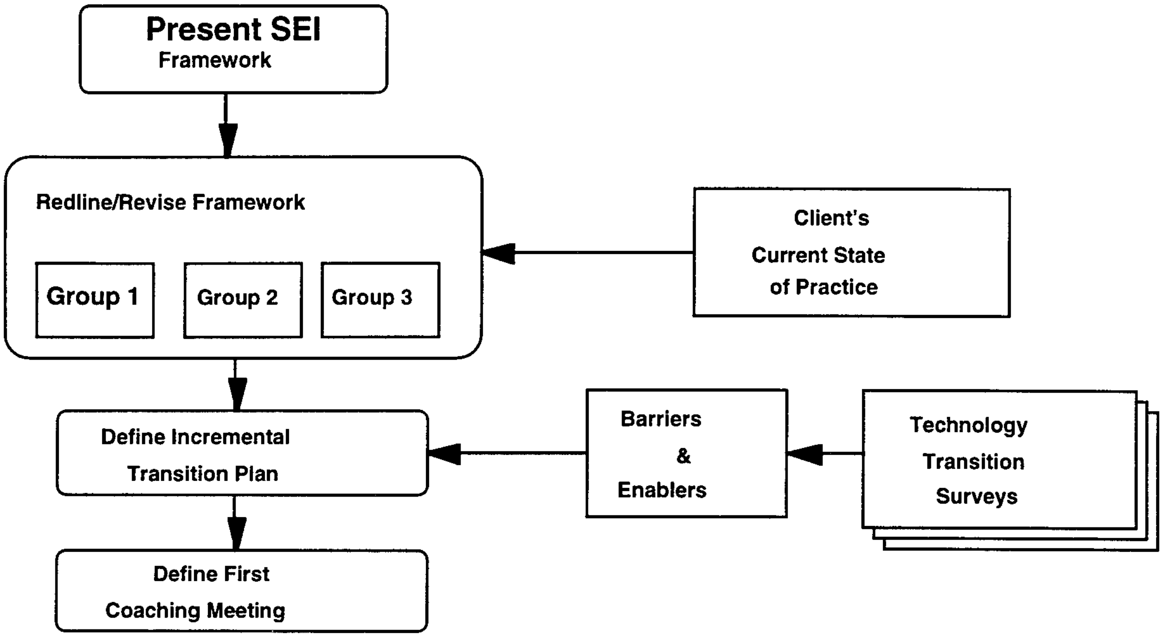


Figure 9: Risk Clinic Process Overview

[1] Examples of such surveys include the following, developed at the SEI by John H. Maher, Jr. and Charles R. Myers, Jr.:

Managing Technological Change: Implementation History Assessment and *Managing Technological Change: Cultural Assessment* (SEI-90-SR-20). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1990.

The above documents are Copyright © IMA 1989. They are available to U.S. government agencies only.

4.2 Supporting Practices

In this group there are three practices: SRE, CRM, and TRM.

4.2.1 Software Risk Evaluation (SRE) Practice

The SRE practice, developed by the SEI, is a formal method for identifying, analyzing, communicating, and mitigating software technical risk [Sisti 94]. It is used by decision makers for evaluating and mitigating the technical risks associated with a software-intensive program or project. The SRE is conducted at major milestones early and periodically in the acquisition life cycle. This practice consists of primary and support functions (see Figure 10). Primary SRE functions are Detection, Specification, Assessment, and Consolidation. The support functions are Planning and Coordination, Verification, and Training and Communication.

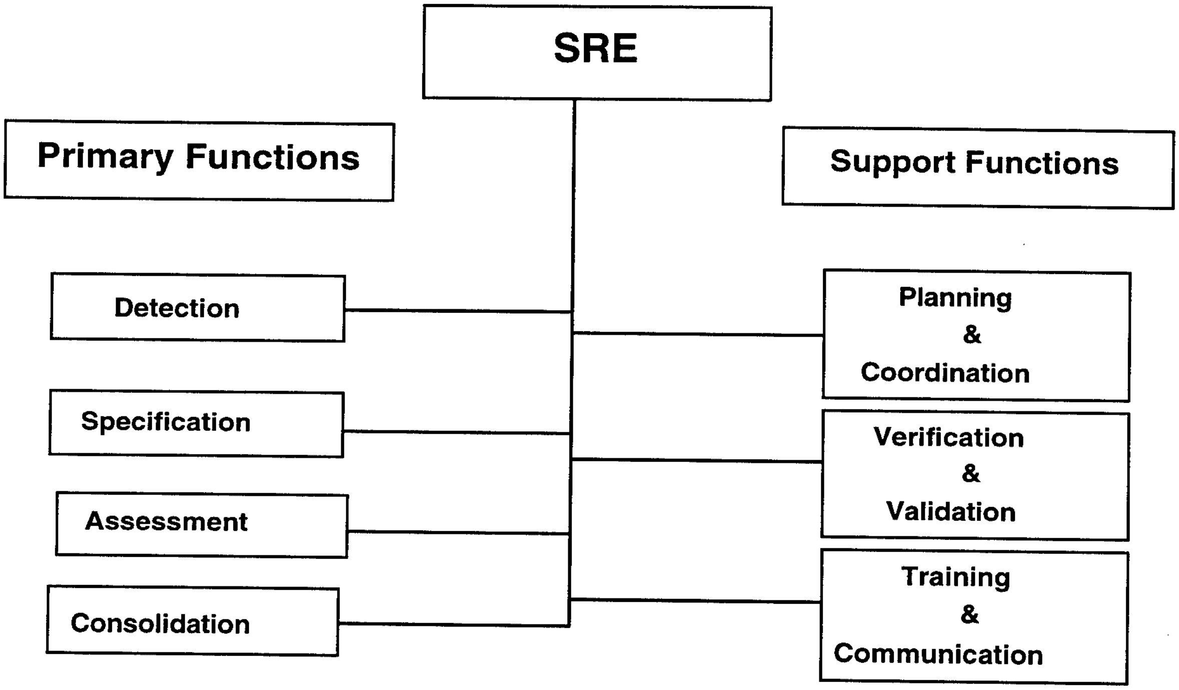


Figure 10: SRE Functional Components

Primary Functions

Four primary functions are identified in the SRE practice—detection, specification, assessment, and consolidation.

Detection is the function of finding software technical risks of a target project. This function ensures systematic and complete coverage of all potential technical risk areas. It also ensures efficiency and effectiveness through the use of appropriate tools and techniques. Risk detection in the SRE practice is performed by using the following:

- SEI Taxonomy-Based Questionnaire ensures complete coverage of all areas of potential software technical risks.
- Selection of appropriate individuals and guidelines for the make-up of the interview groups ensures coverage of all viewpoints including software development and support functions, technicians, and managers.

Risk *specification* is the function of recording all aspects of the identified software technical risk including its condition, consequences, and source. One representation of a software risk statement, (developed, for example, in [Gluch 94]), has several advantages. For instance, it serves as a simple, guiding structure for risk detection activities and for communicating risks coherently and with sufficient detail. It captures components of the risk and simplifies the task of prioritizing, isolating the condition within which the risk applies, and focusing the risk mitigation efforts to the source(s) of the risk. Additionally, risk specification records the source of the particular risk.

Assessment is a function that determines the magnitude of each software technical risk. By definition, magnitude is the product of severity of impact and the probability of an occurrence of the risk.

The SRE practice's mechanism for risk assessment is adapted from previous work conducted by the U.S. Air Force [AFSC 88]. Risk statements are assessed at one of three levels of magnitude—high, medium, or low. The level at which a particular risk is assessed depends on the separate assessments of its severity of impact and its probability of occurrence.

Consolidation is the function of merging, combining, and abstracting risk data into concise chunks of decision-making information. This is necessary due to multiple risk detection activities which identify related risks from different sources. One example is similar risks that are identified in different interview sessions.

Only that set of risk statements which meets the defined criterion are considered as candidates for consolidation. Candidate risk statements must meet *one* of the following criteria for consolidation:

- manifestation of the same risk statement; that is, identical in every way except in the wording of the statements
- fragmentation due to minor variations or different aspects of the same risk statement
- differences in granularity; for example, a minor risk statement which is covered in the context of another risk statement of larger magnitude

4.2.2 Continuous Risk Management (CRM)

CRM is a principle-based practice for managing project risks and opportunities throughout the lifetime of the project. When followed, these principles provide an effective approach to managing risk regardless of the specific methods and tools used. These principles, depicted in Figure 11,[2] are composed of three groups: core, sustaining, and defining.

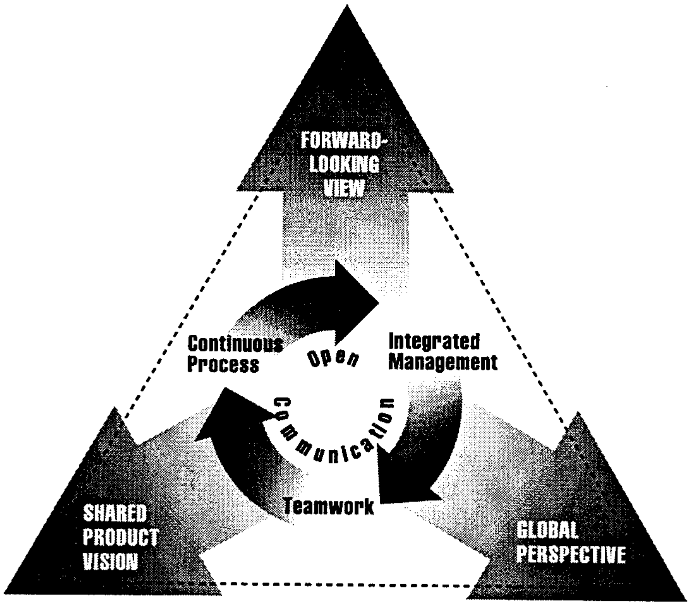


Figure 11: Seven Principles of Risk Management

[2] Williams, Ray C. "Applying the Seven Principles of Team Risk Management." Presented at the Software Engineering Symposium, Pittsburgh, Pa., September 11-14, 1995.

Core Principle

Effective risk management requires constant attention to fostering the *core principle* of open communication. Clearly, the professionals associated with a project are the most qualified to identify the risks in their work on a daily basis. One should always ask, “Does project management provide a conducive environment for staffers to share their concerns regarding potential risks?” or, “Does management follow the ‘killing the messenger’ pattern instead of ‘rewarding the messenger’?” Open communication requires

- encouraging free-flowing information at and between all project levels
- enabling formal, informal, and impromptu communication
- using consensus-based processes that value the individual voice, which can bring unique knowledge and insight to identifying and managing risk

Sustaining Principles

The *sustaining principles* focus on how project risk management is conducted on a daily basis. These are inward-directed, fundamental principles. If established early in the program and constantly nurtured, they should ensure that risk management becomes “the way we do business around here.”

Integrated management

This principle helps to assure that risk management processes, paperwork, and discipline are consistent with established project culture and practice. Risk management is simply an area of emphasis in good project management; therefore, wherever possible, risk management tasks should be integrated into well-established project routine. Integrated management requires

- making risk management an integral and vital part of project management
- adapting risk management methods and tools to a project's infrastructure and culture

Teamwork: No single person can anticipate all the risks that face a project. Risk management requires that project members find, analyze, and work on potential risks together. Group synergy and interdependence in dealing with risk need to be rewarded. Teamwork requires

- working cooperatively to achieve a common goal
- pooling talent, skills, and knowledge

Continuous process: Risk management guidelines must not be allowed to become "shelfware." The processes must be part of daily, weekly, monthly, and quarterly project management. The premise that risk management takes place only during "risk management seasons" is obviously foreign to true management. Continuous process requires

- sustaining constant vigilance
- identifying and managing risks routinely throughout all phases of the project's life cycle

Defining Principles

The *defining principles* focus on how project staff members identify risks, and the extent to which staff and management are ready to address uncertainty. These principles are outward-directed and concerned with focus; they foster the development of shared mental models that clarify the when, why, and what of risk management.

Forward-looking view: This principle develops the ability to look ahead, beyond today's crisis and into the likely consequences and impacts of current decisions on future options. Its staff is also concerned with defining how far into the future to look, so that all risk mitigation efforts of project's staff are complementary. Forward-looking view requires

- thinking toward tomorrow, identifying uncertainties, anticipating potential outcomes
- managing project resources and activities while anticipating uncertainties

Global perspective: This principle requires that project staff replace their parochial views and interests with those that benefit the common good of the overall project. It also demands that the perspectives of the customer be harmonized with those of the supplier to reach a common view of "what's most important to the project." Project staff should develop and share a common viewpoint at a global level, and be able to jointly address and mitigate specific risks. Global perspective requires

- viewing software development within the context of the larger systems-level definition, design, and development
- recognizing both the potential value of opportunity and the potential impact of adverse effects

Shared product vision: This principle focuses on the development of a common understanding of the project's objectives and the goods and services it produces. Once clearly defined, shared product vision makes it much easier to reach a common understanding of what may adversely impact the timeliness, cost, or features of the final result. Shared product vision requires

- sharing a product vision based upon common purpose, shared ownership, and collective commitment
- focusing on results

The functions of CRM, as discussed in Section 4.1.1, are: Identify, Analyze, Plan, Track, Control, and Communicate.

4.2.3 Team Risk Management (TRM)

TRM extends risk management with team-oriented activities involving the customer and supplier (e.g., government and contractor), where both customer and supplier apply the methodologies together [Higuera 94]. TRM establishes an environment built on a set of processes, methods, and tools that enables the customer and supplier to work cooperatively, continuously managing risks throughout the life cycle of a software-dependent development program. It is built on a foundation of the seven principles of risk management discussed in the preface of this paper, and on the philosophy of cooperative teams. Guided by the seven principles, TRM further extends the SEI Risk Management paradigm by adding two functions—initiate and team. Each risk goes through these functions sequentially, but the activity occurs continuously, concurrently, and iteratively throughout the project life cycle (e.g., planning for one risk may identify another). The TRM Guidebook[3] provides an effective instrument with which to familiarize the reader with the concepts, functions, processes, methods, and products of TRM. The guidebook accomplishes this through a description of the overall methodology, a road map for applying it within a project, and detailed descriptions of the processes and methods used to implement the functions of TRM. Figure 12 [Higuera 94] depicts the extension of the SEI Risk Management Paradigm by incorporating the TRM functions (initiate and team)*.

***Initiate**

Recognize the need and commit to create the team culture. Either customer or supplier may initiate team activity, but both must commit to sustain the teams.

***Team** (verb)

Formalize the customer and supplier team and merge the veiwpoints to form a shared product vision. Systematic methods periodically and jointly applied establish a shared understanding of the project risks and their relative importance. Establish joint information base of risks, priorities, metrics, and action plans.

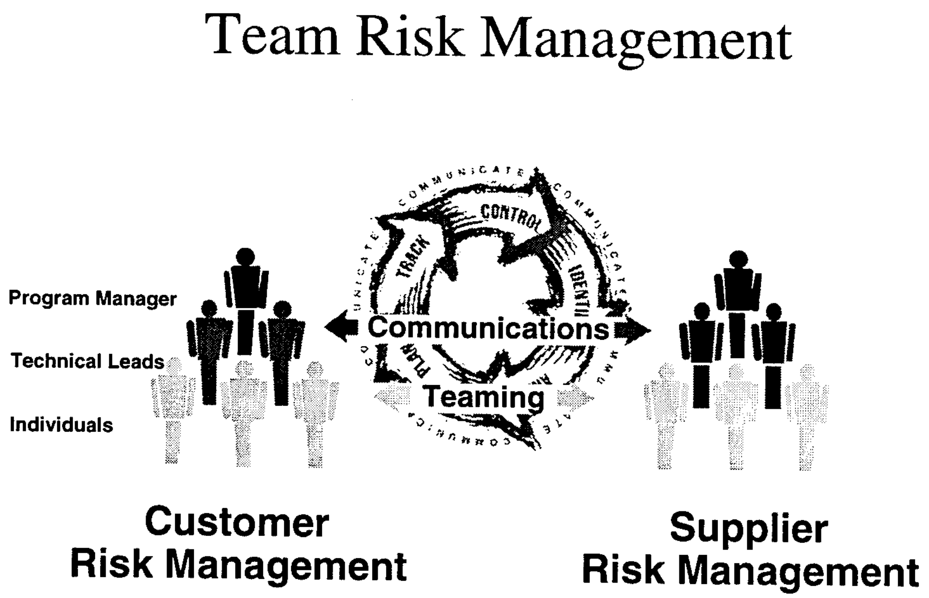


Figure 12: Team Risk Management

[3] Dorofee, A. J., et al. *Team Risk Management Guidebook: Version 0.1.* Software Engineering Institute, Carnegie Mellon University, 1994. Draft technical report not approved for public release.

Note that the last six functions (Identify, Analyze, Plan, Track, Control, and Communicate) are adopted from the risk management paradigm discussed earlier in Section 4.1.1. In summary, TRM offers a number of advantages for a project, as compared to individual risk management. It also involves, however, a change from past customer-supplier (government-contractor) relationships, and this will require new commitments by both. These new commitments, in turn, may involve investment in risk mitigation—particularly early in the program.

4.3 Methodological Framework for Software Risk Management (SRM)

Acquisition and development of large software-driven systems continue to suffer large cost and schedule overruns. While industry is improving its capability and performance through the use of the SW-CMM^{SM} [Humphrey 90] for software, many acquisition organizations continue to operate in an unstable environment. Staffing is based on the availability of individuals, resulting in a random composition of acquisition skills. Very few team members have software acquisition or application domain skills, and little documentation exists to define procedures or capture corporate memory. Software acquisition typically proceeds in an ad hoc manner.

4.3.1 Software Capability Maturity Model (SW-CMM^{SM})

The SW-CMM^{SM} provides software organizations with guidance on how to gain control of their process for developing and maintaining software and how to evolve toward a culture of software engineering excellence. The SW-CMM^{SM} was designed to guide software organizations in selecting process improvement strategies by determining current process maturity and identifying the few issues most critical to software quality and process improvement. By focusing on a limited set of activities and working aggressively to achieve them, organizations can steadily improve their organization-wide software process to enable continuous and lasting gains in software process capability.

The staged structure of the SW-CMM^{SM} is based on product quality principles that have existed for the last 60 years. These principles have been adapted into a maturity framework that establishes the project management and engineering foundation during the initial stages, and quantitatively controls the process during the more advanced stages of maturity.

The maturity framework into which these quality principles have been adapted was first inspired by Philip Crosby in his book *Quality is Free* [Crosby 79]. Crosby's quality management maturity grid describes five evolutionary stages in adopting quality practices. This maturity framework was adapted to the software process by Ron Radice and his colleagues [Radice 85] working under the direction of Watts Humphrey at IBM. Humphrey brought this maturity framework to the SEI in 1986, revised it to add the concept of maturity levels, and developed the foundation for its current use throughout the software industry [Humphrey 90].

4.3.2 Software Acquisition-Capability Maturity Model (SA-CMM^{SM})

The SA-CMM[4] is based upon the principles of the SW-CMM [Humphrey 90]. Similar to the SW-CMM, the SA-CMM describes five levels of organizational software acquisition maturity (see Figure 13). The key process areas (KPAs) define the requirements that must be satisfied in order to accomplish that level of maturity. In other words, progress is made in stages or steps. The levels of maturity and their KPAs thus provide a road map for attaining ever higher levels of maturity.

SA-CMM^{SM} Key Process Areas

Level	Focus	Key Process Areas	
5 Optimizing	*Continuous process improvement*	Acquisition Innovation Management Process Evolution	Quality Productivitity
4 Managed	*Quantitative management*	Quantitative Process Management Quantitative Acquisition Management	
3 Defined	*Acquisition processes and organizational support*	Training Program Software Acquisition Risk Mgt Contract Performance Management Project Performance Management Org Process Defn and Improvement	
2 Repeatable	*Project management processes*	Transition and Maintenance Evaluation Contract Tracking and Oversight Project Office Management Requirements Development and Mgt Solicitation Software Acquisition Planning	
1 Initial	*Competent people and heroics*		Risk Rework

Figure 13: SA-CMM KPAs

The KPAs at any given level describe the minimum requirements for that level of maturity. This does not mean that some portion of those requirements cannot be satisfied or performed at a lower level; in fact, they typically will. However, an organization cannot achieve the next level of maturity unless all the requirements of all lower levels are maintained. The stages of the model are complimentary and flow upward. For example, the tracking and oversight at level 2 will result in corrective actions (reactive approach to defects). This process grows and matures into risk management at level 3 where actions are taken to identify and prepare for risks before

[4] Ferguson, J. J. et al. *Software Acquisition Maturity Model (SA-CMM)* Version 00.02. Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1996. Draft report not publicly released.

they happen (proactive approach). Risk management grows and matures to become defect prevention at level 4 when the potential for a risk is removed by adjusting the process(es) (preventive approach). While defects should decrease as maturity increases, the need for corrective actions (established for level 2) never goes away completely.

The vast experience of the SEI in developing the SW-CMM^{SM} is directly applicable to developing the SA-CMM^{SM}. The two models must be, and are, synergistic. The SW-CMM^{SM} describes the contractor's role in the software acquisition process, while the SA-CMM^{SM} describes the acquirer's role. In addition, the SA-CMM^{SM} includes certain pre-contract-award activities, such as preparing the software statement of work, documentation requirements, and participating in source selection. During the engineering phase of the project, the two models are parallel in their treatment of the processes involved. The SA-CMM^{SM} often ends with the completion of the software acquisition process, when the "new" software is transitioned from the developer to the maintainer. This, however, may not always be the case. In some instances, acquisition offices are being assigned "cradle to grave" responsibility and their authority is being expanded into the maintenance area. In addition, the increased use of incremental and evolutionary deliveries raises a number of maintenance issues *during* the acquisition until the final component of the system is delivered.

The current SW-CMM^{SM} provides the appropriate level of detail for translation into the SA-CMM^{SM}. Ensuring the compatibility of the two models, the items in the SW-CMM^{SM} have been examined and, where appropriate, reworded to reflect the difference between engineering functions (contractor) and the acquisition function (government).

The SA-CMM^{SM} is intended to be generic enough to be used by any organization acquiring software (e.g., subcontracting). For this usage, the term "contractor" refers to the organization, developing the software. The term "government" or "project team" refers to the customer organization and the term "contract" refers to the agreement between the organizations.

Buyers may include a Program Executive Office (PEO) who may be acquiring software across several projects, a Program Manager (PM) who is responsible for a single system acquisition, or a Software Support Activity responsible for supporting PEOs and PMs. The model also includes provisions for the participation of other functional organizations involved, such as testers, product assurance, and laboratories. When conducting a self-assessment, all of these groups should be included, depending on the project organization. The SA-CMM^{SM} does not address the system level acquisition process.

As systems become more complex, a continuing improvement initiative is needed to stay abreast of technology in order to increase efficiency and to take advantage of the latest techniques. There are no road maps, however, to help organizations efficiently improve their technology base and to ensure that they build the best quality products at the lowest cost with the least amount of risk. Although several methodologies exist that can provide an insight into software development and software project management practices, they are not in systematic practice today. The Software Capability Evaluation (SCE) method, for example, does offer a look at organizational capability to produce a product and provides insight into contractor management processes; the SRE method provides a framework for evaluating risks that would prevent project success; and the SW-CMM^{SM} [Humphrey 90] provides through the ranking of the organization's level of technical maturity an index with which to measure the likelihood of success. However, these methods do not explicitly address contractor selection.

If one accepts the premise that more mature software development organizations build better products, then more mature acquisition organizations should be better prepared to do a better job of acquisition. A current argument in the Department of Defense (DoD) is that as DoD acquisition organizations or contractors move from level 1 to level 3, the development organization has to mature as well. Note, for example, that if a level 1 organization is buying from a level 3 organization, the program office might waste time on the wrong issues; it might want to focus on documentation or on reviews, but it does not need that degree of oversight if the level 3 company already does this well.

In order to guide implementation and institutionalization of software acquisition improvement, the SA-CMM^{SM} must be augmented by a framework and road map to guide improvement activities. The Software Acquisition Improvement Framework identifies candidate practices and supporting technologies, expertise, infrastructure, and implementation guidance to satisfy the requirements of KPAs of the SA-CMM^{SM}. The road map shows a path through the possible improvement choices provided by the model, identifying practices for which implementation guidebooks are needed and including measures of the improvement activity's success.

The Acquisition Risk Management Guidebook is one example of a set of guidebooks that will provide practical "how to" practices for selected KPAs. The other guidebooks are: TRM Guidebook, SRM Guidebook, and CRM Guidebook. All these guidebooks are in preparation; they will leverage the lessons learned in risk management and build on earlier work which provided guidance to the source selection process.

5 Deployment of the SEI Risk Management Program

One of the major problems facing software engineering today is the lack of accessible data about development practices and the use of software products. Currently, risk management data are buried within projects and not available to the wider community. Consequently, software engineers are forced to resort to non-empirical arguments in deriving or evaluating many software engineering methods and tools.

The Software Engineering Risk Repository (SERR) is the response of the SEI to this urgent need for an informative database[5]. The SERR is planned to be a national on-line service where widely dispersed information on the development and transfer of software technology will be collected or made available through a variety of sources, including already existing online data-bases, data-gathering instruments such as interviews, questionnaires, reports, and case studies, and printed materials that can be scanned on-line. Technology transfer is a social process which is dependent on the creation of shared meaning and interpretation. Often this is only achievable through sharing trial-and-error experiences with other groups undergoing similar learning and discovery processes. This sharing of experience is an important basis for the construction and dissemination of most software engineering methods, tools, and approaches. The ultimate goal of SERR is to provide a mechanism where the transfer, reception, and evaluation of advanced software engineering process technologies can be communicated, interpreted, and negotiated. The effectiveness of such a mechanism depends on the extent to which relevant and accessible information is made available. In this section, highlights from SEI field work are shared with the reader.

[5] Konda, Suresh L.; Monarch, Ira & Carr, Marvin J. *Software Engineering Risk Repository: Concepts of Operation and Function Requirements.* Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1994. Draft report not publicly released.

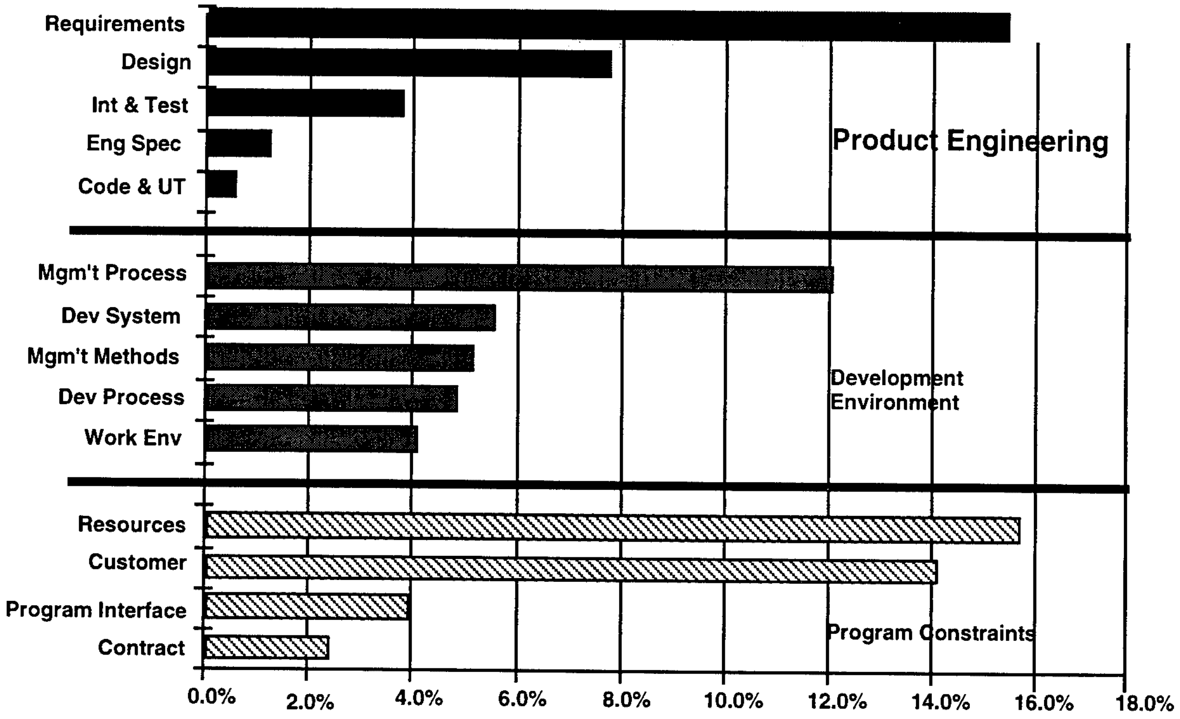


Figure 14: Representation of Levels of Risk From SEI Deployment

This section provides statistical information on the deployment of the risk methodologies that have been developed and deployed by the SEI. The risk data and their analyses have been obtained through the risk assessments and field tests that the SEI conducts as part of its mission. For reasons of propriety and confidentiality, the information is presented here in an aggregate form. As a result of conducting dozens of risk assessments and field tests, the SEI has developed a database on the risks associated with software development.

The usefulness and value of this database are significant and transcend many dimensions:

- It provides important foundations from which new programs can learn and benefit.
- The database represents user and contractor communities with distinct domain knowledge (e.g., airplanes, embedded systems contains applicable information and experiences from different systems and domains). These can be translated, compared, contrasted, and related to new systems and programs (e.g., airplanes to automobiles to submarines; and government to non government).
- The database can be a major asset in supporting the DoD's goal of buying more commercially available systems instead of custom-developed systems.
- The database helps identify risks, and provides linkage between the sources of risk and the appropriate mitigation strategies.

The database also raises some interesting questions, for example: Can we develop a profile of a community from the database?

5.1 Major Classes Within the Hierarchy

Recall that the SEI Taxonomy is built on a hierarchy, with three classes of risk at the highest level: product engineering, development environment, and program constraints. The overall distribution of all the risks in the assessment database within these three classes indicates a surprisingly even division:

- 30% product engineering
- 33% development environment
- 37% program constraints

Below is a summary of the distribution of risks associated with each sub level of the taxonomy hierarchy (see Figure 7). All the raw data were collected and analyzed from the risk assessments conducted by the staff of the SEI Risk Program:

5.2 Major Elements of Risk Within Each Class

Of the five subcategories of risk within product engineering

- Requirements scored over 50% of all risks in the class (at 53%)
- Design scored about 27% of all risks
- Integration and Test scored about 14%

The two remaining categories, Engineering Specialties and Code and Unit Test, scored a total of about 6% (4% and 2%, respectively).

These results are not surprising because they confirm the notion that within product engineering, about 80% of all risks are attributed to Requirements and Design.

Of the five categories of risk within development environment, only Management Process scored appreciably more than the other four categories—about 37%. The remaining four categories—Development System, Management Methods, Development Process, and Work Environment—scored, in descending order, from 17% to 12% respectively. These statistics confirm that management process is critically important in meeting development requirements.

Two categories dominate the sources of risk in the program constraint class: Resources at about 43% and Customer at about 39%. In other words, over 80% of all sources of risk in Program Constraint are attributed to Resources and Customer. The remaining less than 20% are divided between Program Interface at about 11% and Contract at about 7%.

5.3 Major Attributes Within Each Element and Class

5.3.1 Product Engineering Class

Statistical data on each element within the Product Engineering Class are given below:

Requirement element: Among the seven attributes within the Requirement element, *Completeness* dominates the other six attributes at 36%. The remaining attributes scored the following percentages of sources of risk: *Stability* at 21%, *Feasibility* at 14%, *Validity* at 10%, *Precedent* at 8%, *Scale* at 7%, and *Clarity* at 4%.

Design element: The distribution of sources of risk among the six attributes within the Design element decreases gradually as follows: *Non-Developmental Software* at 28%, *Functionality* at 22%, *Performance* at 19%, *Hardware Constraints* at 15%, *Difficulty* at 9%, *Interface* at 7%, and *Testability* at an insignificant level.

Code and Unit element: The sources of risk within the Code and Unit element are uniformly distributed among the three attributes: *Feasibility*, *Testing*, and *Coding/Implementation*.

Integration and Test element: The *Environment* attribute dominates the other two attributes within this element at 72%; *Product Integration* scored 21%, and *System Integration* scored merely 7%.

Engineering Specialties element: The *Specifications attribute* dominates the sources of risk in this element at 58%. The other five attributes scored as follows: *Security* at 25%, *Maintainability* at 9%, *Safety* at 8%, and *Human* factors at an insignificant level.

5.3.2 Development Environment Class

Statistical data on the various elements within the Development Environment Class are given below:

Development Process element: Two attributes dominate the Development Process element: *Formality,* at 48% of all sources of risk within this category, and *Product Control,* at 28%. The remaining 24% are distributed as follows: *Suitability* at 13%, *Familiarity* at 7%, and *Process Control* at 4%. *Deliverability* showed an insignificant level of risk.

Development System element: *Capacity*, *Suitability*, and *Usability* attributes together scored 75% of all sources of risk within this element: *Capacity* at 35%, *Suitability* at 23%, and *Usability* at 17%. The remainder of the six attributes scored as follows: *Familiarity* and *Reliability* each scored 10%, and system support showed an insignificant level of risk.

Management Process element: At 54% the *Planning* attribute dominates all sources of risk within this element. The distribution of the remaining 46% is as follows: *Project Organization* at 24%, *Program Interfaces* at 20%, and *Management Experience* at 2%.

Management Methods element: Over 75% of all risks in this element are related to two attributes: *Personnel Management* at 45% and *Configuration Management* at 33%. Scores for the other attributes were: *Monitoring* 15%, and *Quality Assurance* 7%.

Work Environment element: Of the four attributes in this element, *Communication*, as expected, dominated all others at 74%. The distribution of the risks among the remaining three attributes are as follows: *Quality Attitude* at 24%, and *Cooperation and Morale* at 1% each. These statistics are not surprising; communication in the acquisition process among the user, the customer (often the contracting agent), and the contractor are a major source of risk of cost overrun, time delay in delivery, and risk of not meeting performance criteria.

5.3.3 Program Constraints Class

Statistical data for each element within the Program Constraints Class are given below:

Resources element: At 50%, the *Staff* attribute dominates the other three attributes in this element. The distribution of risk among the remaining attributes is as follows: *Schedule* at 21%, *Facilities* at 18%, and *Budget* at 11%.

Contract element: The distribution of risks among the three attributes within this element is as follows: *Dependencies* at 54%, *Type of Contract* at 36%, and *Restrictions* at 10%.

Program Interfaces element: No attribute dominates this element. *Subcontractors* and *Corporate Management* each scored 25%, *Vendors* scored 22.5%, *Prime Contractor* scored 15%, and *Politics* scored 12.5%. *Associate Contractors* scored an insignificant level of risk.

Customer element: The distribution of risk factors within the seven attributes of the Customer Element ranges from 25% for *Management* to 6% for *Organization.* The remaining attributes scored as follows: *Delays* at 21%, *User Interface* at 19%, *Customer Furnished Resources* at 12%, *Technical Knowledge* at 10%, and *Scope Change* at 6%.

The above statistical data collected by SEI teams sheds important light on some critical sources of risk. *Requirements*, *Management Process*, *Resources*, and *Customer*, for example, are the four most critical sources of risk in software development. Indeed, central to the holistic vision of software risk management depicted in Figure 5 are Needs and Requirements—they determine, to a large extent, the path that software development takes in its evolving life cycle. Another important component in this holistic vision of software risk management is the Human Dimension—Individual, Team, Management, and Stakeholder. People make up the other three critical sources of risk—Management Process, Resources, and Customer. The remaining sources of risk, identified in Figure 14, can be mainly attributed to the Temporal and Methodological dimensions of the holistic vision presented in this paper.

6 Epilogue

This paper presents a brief summary of the methodologies developed by the SEI for the management of risk associated with the acquisition, development, and use of software. Although software continues to grow in importance as a critical system component and, more importantly, as an overall system integrator, major sources of risk the user, the customer, and the contractor communities. The methodologies presented in this paper shed some light on the professional community's effort to assess and ultimately control these inherent risks. Clearly, as systems become increasingly more complex, individual knowledge, judgment, and expertise will not suffice and systemic methodologies for risk management such as those presented in this paper become imperative. This observation, which is based on SEI experience in the deployment of software risk methodologies, is further amplified by the fact that software risk is among the least measured or managed in a system today.

References

[AFSC 88] AFSC/AFLC *Acquisition Management Software Risk Abatement*, Air Force Systems Command and Air Force Logistics Command, Pamphlet 800-45, September 30, 1988.

[Brooks 87] Brooks, Frederick P. "No Silver Bullet," *Computer 20*, 4 (April 1987): 10-19.

[Carr 93] Carr, Marvin J.; Konda, Suresh; Monarch, Ira; Ulrich, Carol; & Walker, Clay. *Taxonomy-Based Risk Identification* (CMU/SEI-93-TR-6, ADA266992). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1993.

[Chittister 93] Chittister, Clyde & Haimes, Yacov Y. "Risk Associated with Software Development: A Holistic Framework for Assessment and Management," *IEEE Transactions on Systems, Man, and Cybernetics 23*, 3 (May-June1993): 710-723.

[Chittister 94] Chittister, Clyde & Haimes, Yacov. "Assessment and Management of Software Technical Risk," *IEEE Transactions on Systems, Man, and Cybernetics 24*, 2 (February 1994): 187-202.

[Crosby 79] Crosby, P.B. *Quality Is Free.* New York: McGraw-Hill, 1979.

[Gluch 94] Gluch, David. *A Construct for Describing Software Development Risks* (CMU/SEI-94-TR-14). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1994.

[Higuera 94] Higuera, Ronald P.; Dorofee, Audrey J.; Walker, Julie A.; & Williams, Ray C. *Team Risk Management: A New Model for Customer-Supplier Relationships* (CMU/SEI-94-SR-005, ADA283987). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1994.

[Haimes 81] Haimes, Yacov Y. "Hierarchical Holographic Modeling," *IEEE Transactions on Systems, Man, and Cybernetics 11*, 9 (September 1981): 606-617. 1981.

[Haimes 91] Haimes, Yacov Y. "Total Risk Management," *Risk Analysis 11*, 2 (June 1991): 169-171.

[Humphrey 90] Humphrey, Watts S. *Managing the Software Process.* New York: Addison-Wesely Publishing Company, Inc., 1990.

[Kaplan 81] Kaplan, S. & Garrick, B. J. "On the Quantitative Definition of Risk," *Risk Analysis 1*, 1 (March 1981): 11-27.

[Katzenbach 93] Katzenbach, Jon R. & Smith, Douglas K. *The Wisdom of Teams.* New York: Harper Business, 1993.

[Kirkpatrick 92] Kirkpatrick, Robert J.; Walker, Julie; & Firth, Robert. "Software Development Risk Management: An SEI Appraisal," *Software Engineering Institute Technical Review '92* (CMU/SEI-92-REV). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1992.

[Lowrance 76] Lowrance, William W. *Of Acceptable Risk: Science and the Determination of Safety.* Los Altos, Ca: William Kaufmann, 1976.

[Sisti 94] Sisti, Francis J. & Joseph, Sujoe. *Software Risk Evaluation Method* CMU/SEI-94-TR-19). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1994.

[Van Scoy 92] Van Scoy, Roger L. Software Development Risk: Opportunity, Not Problem (CMU/SEI-92-TR-30, ADA 258743). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1992

[House 89] United States House of Representatives Committee on Science, Space, and Technology, Subcommittee on Investigations and Oversight. *Bugs in the Program: Problems in the Federal Government Computer Software Development and Regulation.* Washington, D.C.: United States Government Printing Office, 1989.

UNLIMITED, UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS
None

2a. SECURITY CLASSIFICATION AUTHORITY
N/A

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE
N/A

3. DISTRIBUTION/AVAILABILITY OF REPORT
Approved for Public Release
Distribution Unlimited

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
CMU/SEI-96-TR-012

5. MONITORING ORGANIZATION REPORT NUMBER(S)
ESC-TR-96-012

6a. NAME OF PERFORMING ORGANIZATION
Software Engineering Institute

6b. OFFICE SYMBOL (if applicable)
SEI

7a. NAME OF MONITORING ORGANIZATION
SEI Joint Program Office

6c. ADDRESS (city, state, and zip code)
Carnegie Mellon University
Pittsburgh PA 15213

7b. ADDRESS (city, state, and zip code)
HQ ESC/ENS
5 Eglin Street
Hanscom AFB, MA 01731-2116

8a. NAME OFFUNDING/SPONSORING ORGANIZATION
SEI Joint Program Office

8b. OFFICE SYMBOL (if applicable)
ESC/ENS

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
F19628-95-C-0003

8c. ADDRESS (city, state, and zip code))
Carnegie Mellon University
Pittsburgh PA 15213

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO	PROJECT NO.	TASK NO	WORK UNIT NO.
63756E	N/A	N/A	N/A

11. TITLE (Include Security Classification)
Software Risk Management

12. PERSONAL AUTHOR(S)
Ron Higuera, Yacov P. Haimes

13a. TYPE OF REPORT
Final

13b. TIME COVERED
FROM TO

14. DATE OF REPORT (year, month, day)
June 1996

15. PAGE COUNT
48

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (continue on reverse of necessary and identify by block number)
continuous risk management, software risk evaluation
software risk management, team risk management

19. ABSTRACT (continue on reverse if necessary and identify by block number)

This paper presents a holistic vision of the risk-based methodologies for Software Risk Management (SRM) developed at the Software Engineering Institute (SEI). SRM methodologies address the entire life cycle of software acquisition, development, and maintenance. This paper is driven by the premise that the ultimate efficacy of the developed methodologies and tools for software engineering is to buy smarter, manage more effectively, identify opportunities for continuous improvement, use available information and databases more efficiently, improve industry, raise the community's playing field, and review and evaluate progress. The methodologies are based on seven management principles: shared product vision, teamwork, global perspective, forward-looking view, open communication, integrated management, and continuous process.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
UNCLASSIFIED/UNLIMITED ■ SAME AS RPT □ DTIC USERS ■

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified, Unlimited Distribution

22a. NAME OF RESPONSIBLE INDIVIDUAL
Thomas R. Miller, Lt Col, USAF

22b. TELEPHONE NUMBER (include area code)
(412) 268-7631

22c. OFFICE SYMBOL
ESC/ENS (SEI)

EDITION of 1 JAN 73 IS OBSOLETE UNLIMITED, UNCLASSIFIED

ABSTRACT — continued from page one, block 19